REMARKS

The Examiner has rejected Claims 1-36 under 35 U.S.C. 112, first paragraph, as based on a disclosure which is allegedly not enabling.

First, the Examiner asks "[w]hat is the emulation extension in the present context?" In response, applicant quotes the claims which indicate that "the first emulator extension include[es] program instructions that aid in the process of emulating the suspect code in order to detect a computer virus and/or malicious software." Moreover, applicant points to the originally filed description where an enabling description of the extension(s) is provided during the description of Figures 2 and 3 on pages 7-10.

Moreover, the Examiner asks "[w]hy the emulation extension [is] need[ed], and how the extensions relate to the emulator?" Again, applicant quotes the claims which indicate that "the first emulator extension causes the emulator to examine the suspect code looking for patterns that indicate that the suspect code is likely to exhibit malicious behavior." Moreover, applicant points to the originally filed description where an enabling description is provided during the description of Figures 2 and 3 on pages 7-10.

The Examiner rejects Claims 1-36 under 35 U.S.C. 103(a) as being unpatentable over Bond et al., U.S. Patent No. 6,275,938. Applicant respectfully disagrees with this rejection, especially in view of the amendments made hereinabove.

For example, the Examiner simply dismisses, without a specific prior art showing, applicant's claimed "determining whether the suspect code is likely to exhibit malicious behavior based upon the emulation" (see all independent claims). Applicant respectfully disagrees with such assertion, as Bond merely "prevent[s] breaches," and makes absolutely no determination as to "whether the suspect code is likely to exhibit malicious behavior based upon the emulation." Only applicant teaches and claims such a determination operation, which enables "detect[ion of] a computer virus and/or malicious software within suspect code," as claimed (emphasis added).

To establish a *prima facie* case of obviousness, three basic criteria must be met. First, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed.Cir.1991).

Applicant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations. Since the prior art references, when combined, fail to teach or suggest all the claim limitations, it appears that the Examiner has relied on an inherency/obviousness argument regarding the above emphasized claim limitations. In view of the arguments made hereinabove, any such inherency/obviousness argument has been adequately rebutted, and a notice of allowance or a specific prior art showing of such claim features, in combination with the remaining claim elements, is respectfully requested.

Further, the Examiner relies on the following excerpt from Bond to make a prior art showing of applicant's claimed "loading a first emulator extension into the emulator, the first emulator extension including program instructions that aid in the process of emulating the suspect code in order to detect a computer virus and/or malicious software" (see all independent claims).

> "Step 421 substitutes the applet's static links with links to thunk modules. That is, emulator 39 finds all calls to APIs 352-354 within the code of applet 362 and changes them to calls to the corresponding thunks 391-393. A static link is a link that remains constant during the execution of the applet. A DLL, or Dynamic Link Library, is a library of executable functions or data that can be used by a Windows application. Typically, a DLL provides one or more particular functions and a DLL is accessed by creating either static or dynamic link to the DLL. DLL files in the following description end with the extension .dll. A thunk DLL is a secure API within the sandbox. The thunk DLLs block or restrict many APIs that are not considered safe. For instance, CreateFile will be allowed only in known locations. Similarly, an applet will not be allowed to create another

process to record passwords. As described above, some thunks merely pass control to the corresponding API. For example, the Win32 APIs named "CreateWindow", "CreateDialog", "CreateIcon", "CreateCursor", and similar functions do not affect other processes, and can be permitted to untrusted code. On the other hand, certain other APIs must be made entirely unavailable to untrusted code. For example, allowing "CreateProcess" would allow the untrusted applet to run another program outside the sandbox; operations such as "ExitWindowsEx( )" are blocked completely, so that the untrusted code cannot log-off the current user or power-down the computer. A thunk such as 393 blocks an API by returning an error code back to the control, as symbolized by line 395." (see col. 6, lines 24-52)

Such excerpt and the remaining Bond reference, however, merely suggest thunk DLL extensions which operate to block or restrict APIs that are not considered safe. There is simply no disclosure, teaching or even suggestion of any sort of an extension that "aid[s] in the process of emulating the suspect code in order to detect a computer virus and/or malicious software." Only applicant teaches and claims such an emulation/detection-aiding extension, as claimed.

Again, applicant respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations.

Despite this clear distinction already present in the claims and in the spirit of expediting the prosecution of the present application, applicant now claims in each of the independent claims:

> "loading a first emulator extension into the emulator, the emulator capable of performing an emulation using emulation code, the first emulator extension including program instructions that aid in the process of emulating the suspect code in order to detect a computer virus and/or malicious software, wherein the program instructions of the first emulator extension are additional beyond that associated with the emulator code, for assisting the emulator code in the emulation by patching the additional program instructions into the emulator in order to aid in detecting the computer virus and/or malicious software within the suspect code" (emphasis added - see this or similar language in each of the independent claims).

By these amendments, applicant had emphasizes the functionality of applicant's claimed extension as code that operates as a patch for providing additional program instructions into the emulator in order to aid in detecting the computer virus and/or malicious software within the suspect code. Only applicant teaches and claims such an emulator extension, as claimed.

Applicant further notes that the Examiner's application of the prior art to the dependent claims is also replete with deficiencies. Just by way of example, with respect to Claim 2 et al., the Examiner simply argues, without a specific prior art showing, that Bond discloses applicant's claimed "wherein loading the first emulator extension into the emulator includes loading the first emulator extension into the emulator buffer within the emulator; and wherein performing the emulation includes emulating the program instructions that comprise the first emulator extension" (see Claim 2 et al.).

The Bond reference, however, makes absolutely no suggestion of applicant's claimed "wherein loading the first emulator extension into the emulator includes loading the first emulator extension into the emulator buffer within the emulator; and wherein performing the emulation includes emulating the program instructions that comprise the first emulator extension" (see Claim 2 et al.).

Further, the Examiner relies on col. 4, lines 39-57, col. 5, lines 23-33 from Bond to make a prior art showing of applicant's claimed "emulating the suspect code prior to loading the first emulator extension into the emulator buffer" (see Claim 5 et al.). Such excerpt and the remaining Bond reference, however, makes absolutely no suggestion regarding the order of the emulation and loading of the emulator extension into an emulation buffer, as claimed by applicant. Only applicant teaches and claims emulation prior to first emulation extension loading, as claimed.

With respect to Claim 6 et al., the Examiner simply argues, without a specific prior art showing, that Bond discloses applicant's claimed "loading a second emulator extension into the emulator; and performing a second emulation using the second emulator extension and the suspect code" (see Claim 6 et al.). However, the Bond reference makes absolutely no suggestion of

applicant's claimed second emulator extension, by virtue of the fact that there is no first emulator extension, as noted hereinabove.

With respect to Claim 7 et al., the Examiner simply argues, without a specific prior art showing, that Bond discloses applicant's claimed "wherein the first emulator extension and the second emulator extension provide support for conflicting emulator environments" (see Claim 7 et al.). The Bond reference, however, makes absolutely no suggestion of emulating in conflicting emulator environments, let alone a first and second emulator extension for attending to such conflicting emulator environments, as claimed.

With respect to Claim 9 et al., the Examiner simply argues, without a specific prior art showing, that Bond discloses applicant's claimed "wherein the first emulator extension includes code for decrypting an encrypted computer virus and other encrypted malicious code" (see Claim 9 et al.). The Bond reference, however, makes absolutely no suggestion of applicant's claimed first emulator extension, let alone a first emulator extension that includes code for decrypting an encrypted computer virus and other encrypted malicious code. It is even noted that Bond does not even address encryption, in general, for that matter.

With respect to Claim 10 et al., the Examiner simply argues, without a specific prior art showing, that Bond discloses applicant's claimed technique whereby "if a computer virus or other malicious software is detected within the suspect code, disinfecting the suspect code" (see Claim 10 et al.). However, the Bond reference makes absolutely no suggestion of code disinfection, but rather stops short at security breach prevention, etc.

Applicant again respectfully asserts that at least the third element of the *prima facie* case of obviousness has not been met, since the prior art references, when combined, fail to teach or suggest all the claim limitations. Again, a notice of allowance or a specific prior art showing of such claimed features, in combination with the remaining claim limitations, is respectfully requested.

Still yet, applicant brings to the Examiner's attention the subject matter of new Claim 27 below, which is deemed to be novel:

"wherein if the computer virus and/or malicious software is not detected, it is determined if there are any remaining emulator extensions remaining in a database that have not already been used, wherein if there are any remaining emulator extensions remaining in the database that have not already been used, a next emulator extension is loaded into the emulator to repeat the emulation using the next emulator extension."

A notice of allowance or a specific prior art showing of such claimed features, in combination with the remaining claim limitations, is respectfully requested.

Thus, all of the independent claims are deemed allowable. Moreover, the remaining dependent claims are further deemed allowable, in view of their dependence on such independent claims.

In the event a telephone conversation would expedite the prosecution of this application, the Examiner may reach the undersigned at (408) 505-5100. The Commissioner is authorized to charge any additional fees or credit any overpayment to Deposit Account No. 50-1351 (Order No. NAI1P268/99.081.01).

Respectfully submitted,
Zilka-Kotab, PC.

Kevin J. Zilka
Registration No. 41,429

P.O. Box 721120
San Jose, CA 95172-1120
408-505-5100